SIAM J. DISCRETE MATH.
Vol. 33, No. 3, pp. 1194–1215
© 2019 Society for Industrial and Applied Mathematics

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

PACKING CYCLES FASTER THAN ERDOS–POSA*

DANIEL LOKSHTANOV†, AMER E. MOUAWAD‡, SAKET SAURABH§, AND MEIRAV ZEHAVI¶

Abstract. The CYCLE PACKING problem asks whether a given undirected graph $G = (V, E)$ contains k vertex-disjoint cycles. Since the publication of the classic Erdős–Pósa theorem in 1965, this problem received significant attention in the fields of graph theory and algorithm design. In particular, this problem is one of the first problems studied in the framework of parameterized complexity. The nonuniform fixed-parameter tractability of CYCLE PACKING follows from the Robertson–Seymour theorem, a fact already observed by Fellows and Langston in the 1980s. In 1994, Bodlaender showed that CYCLE PACKING can be solved in time $2^{\mathcal{O}(k^2)} \cdot |V|$ using exponential space. In the case a solution exists, Bodlaender's algorithm also outputs a solution (in the same time). It has later become common knowledge that CYCLE PACKING admits a $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time (deterministic) algorithm using exponential space, which is a consequence of the Erdős–Pósa theorem. Nowadays, the design of this algorithm is given as an exercise in textbooks on parameterized complexity. Yet, no algorithm that runs in time $2^{o(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, has been found. In light of this, it seems natural to ask whether the $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$ bound is essentially optimal. In this paper, we answer this question negatively by developing a $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$-time (deterministic) algorithm for CYCLE PACKING. In the case a solution exists, our algorithm also outputs a solution (in the same time). Moreover, apart from beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, our algorithm runs in time linear in $|V|$, and its space complexity is polynomial in the input size.

Key words. parameterized complexity, graph algorithms, cycle packing, Erdős–Pósa theorem

AMS subject classifications. 05C85, 68Q17, 68Q25, 68W05, 68W25, 68W40

DOI. 10.1137/17M1150037

1. Introduction.

The CYCLE PACKING problem asks whether a given undirected graph $G = (V, E)$ contains k vertex-disjoint cycles. Since the publication of the classic Erdős–Pósa theorem in 1965 [16], this problem received significant attention in the fields of graph theory and algorithm design. The problem is known to be NP-complete as it contains partition into triangles as a special case (take $k = |V|/3$) [21]. Furthermore, CYCLE PACKING is one of the first problems studied in the framework of parameterized complexity. In this framework, each problem instance is associated with a parameter k that is a nonnegative integer, and a problem is said to be *fixed-parameter tractable (FPT)* if the combinatorial explosion in the time complexity can be confined to the parameter k. More precisely, a problem is FPT if it can be solved in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some function f, where $|I|$ is the input size. For more information, we refer the reader to recent monographs such as [15, 11, 19].

*Received by the editors October 2, 2017; accepted for publication (in revised form) May 6, 2019; published electronically July 2, 2019. A preliminary version of this paper appeared in *Proceedings of ICALP'17*.
https://doi.org/10.1137/17M1150037
Funding: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant 306992.
†University of Bergen, Norway (daniello@ii.uib.no).
‡American University of Beirut, Lebanon (aa368@aub.edu.lb).
§University of Bergen, Norway, and Institute of Mathmatical Sciences, HBNI, India (saket@imsc.res.in).
¶Ben-Gurion University, Israel (meiravze@bgu.ac.il).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

In this paper, we study the Cycle Packing problem from the perspective of parameterized complexity. In the standard parameterization of Cycle Packing, the parameter is the number k of vertex-disjoint cycles. The nonuniform fixed-parameter tractability of Cycle Packing follows from the Robertson–Seymour theorem [41],[1] a fact already observed by Fellows and Langston in the 1980s. In 1994, Bodlaender showed that Cycle Packing can be solved in time $2^{\mathcal{O}(k^2)} \cdot |V|$ using exponential space [3]. Notably, in the case a solution exists, Bodlaender's algorithm also outputs a solution in time $2^{\mathcal{O}(k^2)} \cdot |V|$.

The Erdős–Pósa theorem states that there exists a function $f(k) = \mathcal{O}(k \log k)$ such that for each nonnegative integer k, every undirected graph either contains k vertex-disjoint cycles or it has a feedback vertex set consisting of $f(k)$ vertices [16]. It is well known that the treewidth (tw) of a graph is not larger than its feedback vertex set number (fvs) and that a naive dynamic programming (DP) scheme solves Cycle Packing in time $2^{\mathcal{O}(tw \log tw)} \cdot |V|$ and exponential space (see, e.g., [11]). Thus, the existence of a $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time (deterministic) algorithm that uses exponential space can be viewed as a direct consequence of the Erdős–Pósa theorem. Nowadays, the design of this algorithm is given as an exercise in textbooks on parameterized complexity such as [15] and [11]. In the case a solution exists, this algorithm does not output a solution (though we remark that with a certain amount of somewhat nontrivial work, it is possible to modify this algorithm to also output a solution).

Prior to our work, no algorithm that runs in time $2^{o(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, has been found. In light of this, it seemed tempting to ask whether the $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$ bound is essentially optimal. In particular, two natural directions to explore in order to obtain a faster algorithm necessarily lead to a dead end. First, Erdős and Pósa [16] proved that the bound $f(k) = \mathcal{O}(k \log k)$ in their theorem is essentially tight as there exist infinitely many graphs and a constant c such that these graphs do not contain k vertex-disjoint cycles and yet their feedback vertex set number is larger than $ck \log k$. Second, Cygan et al. [12] proved that the bound $2^{\mathcal{O}(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ is also likely to be essentially tight in the sense that unless the exponential time hypothesis (ETH) [23] is false, Cycle Packing cannot be solved in time $2^{o(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$. (However, it might still be true that Cycle Packing is solvable in time $2^{o(fvs \log fvs)} \cdot |V|^{\mathcal{O}(1)}$.) With respect to the parameter k, one can show that Cycle Packing does not admit an algorithm with running time $2^{o(k)} \cdot |V|^{\mathcal{O}(1)}$, unless ETH fails. This follows from the fact that the problem of partitioning into triangles does not admit an algorithm with running time $2^{o(|V|)} \cdot |V|^{\mathcal{O}(1)}$, unless ETH fails, even when input graph is of degree at most 4 [44].

1.1. Related work. The Cycle Packing problem admits a factor $\mathcal{O}(\log |V|)$ approximation algorithm [33], and it is quasi-NP-hard to approximate within a factor of $\mathcal{O}(\log^{\frac{1}{2}-\epsilon} |V|)$ for any $\epsilon > 0$ [20]. In the context of kernelization with respect to the parameter k (a polynomial time algorithm that given an instance of the problem returns an equivalent instance of the problem, with the size and the parameter of the output instance being bounded by a function of the input parameter), Cycle Packing does not admit a polynomial kernel unless NP $\subseteq$ coNP/Poly [6]. We refer to [11] for the notion of kernelization and for the theory of lower bounds on kernelization. Recently, Lokshtanov et al. [34] defined a notion of "approximate kernelization" and obtained a 6-approximate kernel with $\mathcal{O}((k \log k)^2)$ vertices along with a $(1+\epsilon)$-approximate kernel with $k^{\mathcal{O}(f(\epsilon))}$ vertices for some function f. We would like to

[1]The paper [41] was already available as a manuscript in 1986 (see, e.g., [3]).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

mention that in the case one seeks k edge-disjoint cycles rather than k vertex-disjoint cycles, the problem becomes significantly simpler in the sense that it admits a kernel with $\mathcal{O}(k \log k)$ vertices [6].

Focusing on structural parameters, Bodlaender, Jansen, and Kratsch [4] obtained polynomial kernels with respect to the vertex cover number, vertex-deletion distance to a cluster graph, and max leaf number. In planar graphs, Bodlaender, Penninkx, and Tan [5] solved Cycle Packing in subexponential time $2^{\mathcal{O}(\sqrt{k})} \cdot |V|^{\mathcal{O}(1)}$ and showed that this problem admits a linear kernel. In the more general class of H-minor-free graphs, Dorn, Fomin, and Thilikos [14] also solved Cycle Packing in subexponential time $2^{\mathcal{O}(\sqrt{k})} \cdot |V|^{\mathcal{O}(1)}$. Moreover, for apex-minor-free graphs, Fomin et al. [18] showed that Cycle Packing admits a linear kernel, and Fomin et al. [17] showed that it also admits an EPTAS. When the input graph is a directed graph, Cycle Packing is W[1]-hard [42], but it admits an FPT approximation scheme [22]. In fact, Cycle Packing in directed graphs was the first W[1]-hard problem shown to admit such a scheme. Krivelevich et al. [33] obtained a factor $\mathcal{O}(|V|^{\frac{1}{2}})$ approximation algorithm for Cycle Packing in directed graphs and showed that this problem is quasi-NP-hard to approximate within a factor of $\mathcal{O}(\log^{1-\epsilon}|V|)$ for any $\epsilon > 0$.

Several variants of Cycle Packing have also received significant attention. For example, the variant of Cycle Packing where one seeks k *odd* vertex-disjoint cycles has been widely studied [39, 43, 38, 32, 30, 31]. Another well-known variant, where the cycles need to contain a prescribed set of vertices, has also been extensively investigated [27, 36, 28, 26, 29]. Furthermore, a combination of these two variants has been considered in [26, 25].

Finally, we briefly mention that inspired by the Erdős–Pósa theorem, a class of graphs $\mathcal{H}$ is said to have the Erdős–Pósa property if there is a function $f(k)$ for which given a graph G, it either contains k vertex-disjoint subgraphs such that each of these subgraphs is isomorphic to a graph in $\mathcal{H}$, or it contains a set of $f(k)$ vertices that hits each of its subgraphs that is isomorphic to a graph in $\mathcal{H}$. A fundamental result in graph theory by Robertson and Seymour [40] states the the class of all graphs that can be contracted to a fixed planar graph H has the Erdős–Pósa property. Recently, Chekuri and Chuzhoy [7] presented a framework that leads to substantially improved functions $f(k)$ in the context of results in the spirit of the Erdős–Pósa theorem. Among other results, these two works are also related to the recent breakthrough result by Chekuri and Chuzhoy [8], which states that every graph of treewidth at least $f(k) = \mathcal{O}(k^{98} \cdot \mathrm{polylog}(k))$ contains the $k \times k$ grid as a minor. (The constant 98 has been improved to 36 in [9] and to 19 in [10].) Following the seminal work by Robertson and Seymour [40], numerous papers (whose survey is beyond the scope of this paper) investigated which other classes of graphs have the Erdős–Pósa property, which are the "correct" functions f associated with them, and which generalizations of this property lead to interesting discoveries.

1.2. Our contribution. In this paper, we show that the running time of the algorithm that is a consequence of the Erdős–Pósa theorem is not essentially tight. For this purpose, we develop a $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$-time (deterministic) algorithm for Cycle Packing. In the case a solution exists, our algorithm also outputs a solution (in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$). Moreover, apart from beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, our algorithm runs in time linear in $|V|$, and its space complexity is polynomial in the input size. Thus, we also improve upon the classical $2^{\mathcal{O}(k^2)} \cdot |V|$-time algorithm by Bodlaender [3]. Our result is summarized in the following theorem.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

THEOREM 1.1. *There exists a (deterministic) polynomial-space algorithm that solves* CYCLE PACKING *in time* $2^{\mathcal{O}(\frac{k\log^2 k}{\log\log k})} \cdot |V|$. *In case a solution exists, it also outputs a solution.*

On a high level, to obtain the dependency on k specified in Theorem 1.1, our approach revisits the classic algorithm based on the Erdős–Pósa property, and to perform some trade-off between the size of the modulator and the structure of the graph obtained by removing it. Specifically, the idea is to compute a "relaxed feedback vertex set": a set S of size $\mathcal{O}(k\log k/\log\log k)$ so that $G - S$ is not necessarily a forest, but it has no short cycles (of length $\mathcal{O}(\log k/\log\log k)$) after contracting long detached paths to single edges. This creates a "relaxed" modulator that is smaller by a $\mathcal{O}(\log\log k)$ multiplicative factor than if we required S to be a feedback vertex set. After the application of reductions that further simplify $G - S$, and by using the relaxed modulator, one can roughly guess the interaction between S and $G - S$ in the solution; this step amounts to guessing one of around $|S|!$ options, which costs $\mathcal{O}(2^{k\log^2 k/\log\log k})$. Having fixed the interaction, we can contract any remaining long paths so that the whole remaining graph has $\mathcal{O}(k\log^{3/2} k)$ vertices; it is important here that this step goes through even when we work with the "relaxed" modulator S as explained above, instead of a normal feedback vertex set as in the classic algorithm. As the size of the graph is already bounded, a simple dynamic programming on subsets suffices to finish the proof.

More broadly, our technique relies on several combinatorial arguments that might be of independent interest, and whose underlying ideas might be relevant to the design of other parameterized algorithms. Let us now outline the structure of our proof, specifying the main ingredients that we develop along the way.

- First, we show that in time linear in $|V|$, it is easy to bound $|E|$ by $\mathcal{O}(k\log k \cdot |V|)$ (Assumption 1).
- Second, we give an algorithmic version of the Erdős–Pósa theorem that runs in time linear in $|V|$ and which outputs either a solution or a small feedback vertex set (Theorem 3.2).
- Then, we show that given a graph $G = (V, E)$ and a feedback vertex set F, a shortest cycle in G can be found in time $\mathcal{O}(|F| \cdot (|V| + |E|))$ (Lemma 3.5).
- We proceed by interleaving an application of a simple set of reduction rules (Reduction Rules A1, A2, and A3) with a computation of a "short" cycle. Given some $g > 6$, we obtain a set S of size smaller than gk such that the girth of the "irreducible component" of $G - S$ is larger than g (Lemma 4.2). Here, the irreducible component of $G - S$ is the graph obtained from $G - S$ by applying our reduction rules.
- Next, we show that the number of vertices in the above-mentioned irreducible component is actually "small"—for some fixed constant c, it can be bounded by $(2ck\log k)^{1+\frac{6}{g-6}} + 3ck\log k$ (Lemma 4.3). The choice of $g = \frac{48\log k}{\log\log k} + 6$ results in the bound $3ck\log k + 2ck\log^{1.5} k$ (Corollary 4.4).[2]
- Now, we return to examine the graph $G - S$ rather than only its irreducible component. The necessity of this examination stems from the fact that our reduction rules, when applied to $G - S$ rather than G, do not preserve solutions. We first give a procedure which given any set X modifies the graph

[2]We found these constants as the most natural ones to obtain a clean proof of *any* bound of the form $\mathcal{O}(\frac{k\log^2 k}{\log\log k})$ (that is, the constants were not optimized to obtain the bound $3ck\log k + 2ck\log^{1.5} k$).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

$G-X$ in a way that both preserves solutions and gets rid of many leaves (Lemma 5.1). We then use this procedure to bound the number of leaves, as well as other "objects," in the reducible component of $G-S$ (Lemma 5.2).

- At this point, the graph G may still contain many vertices: the reducible component of $G-S$ may contain "long" induced paths (which are not induced paths in G). We show that the length of these paths can be shortened by "guessing" permutations that provide enough information describing the relations between these paths and the vertices in S. Overall, we are thus able to bound the entire vertex-set of G by $\mathcal{O}(k \log^{1.5} k)$ in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log\log k})} \cdot |V|$ and polynomial space (Lemma 6.1).
- Finally, we apply a DP scheme (Lemma 7.1). Here, to ensure that the space complexity is polynomial in the input size, we rely on the principle of inclusion-exclusion.

2. Preliminaries.

We use standard terminology from the book of Diestel [13] for those graph-related terms that are not explicitly defined here. We only consider finite graphs possibly having self-loops and multiedges. Moreover, using an appropriate reduction rule we restrict the maximum multiplicity of an edge to be 2. For a graph G, we use V and E to denote the vertex and edge sets of the graph G, respectively. For a vertex $v \in V$, we use $\deg_G(v)$ to denote the degree of v, i.e., the number of edges incident on v, in the (multi) graph G. We also use the convention that a self-loop at a vertex v contributes 2 to its degree. For a vertex subset $S \subseteq V$, we let $G[S]$ and $G-S$ denote the graphs induced on S and $V \setminus S$, respectively. For a vertex subset $S \subseteq V$, we use $N_G(S)$ and $N_G[S]$ to denote the open and closed neighborhoods of S in G, respectively. That is, $N_G(S) = \{v \mid \{u,v\} \in E, u \in S\} \setminus S$ and $N_G[S] = N_G(S) \cup S$. In the case $S = \{v\}$, we simply let $N(v) = N(S)$ and $N[v] = N[S]$. For a graph $G = (V,E)$ and an edge $e \in E$, we let G/e denote the graph obtained by contracting e in G. For $E' \subseteq \binom{V}{2}$, i.e., a subset of edges, we let $G + E'$ denote the (multi) graph obtained after adding the edges in E' to G, and we let G/E' denote the (multi) graph obtained after contracting the edges of E' in G. The girth of a graph is denoted by girth(G), its minimum degree by $\delta(G)$, and its maximum degree by $\Delta(G)$. A graph with no cycles has infinite girth.

A *path* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $\{v_i, v_{i+1}\}$ is an edge for all $0 \leq i < \ell$. A *cycle* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $\{v_i, v_{(i+1) \mod \ell+1}\}$ is an edge for all $0 \leq i \leq \ell$. Both a double edge and a self-loop are cycles. If P is a path from a vertex u to a vertex v in the graph G, then we say that u and v are the end vertices of the path P and P is a (u,v)-path. For a path P, we use $V(P)$ and $E(P)$ to denote the sets of vertices and edges in the path P, respectively, and a length of P is denoted by $|P|$ (i.e, $|P| = |V(P)|$). For a cycle C, we use $V(C)$ and $E(C)$ to denote the sets of vertices and edges in the cycle C, respectively, and the length of C, denoted by $|C|$, is $|V(C)|$. For a path or a cycle Q we use $N_G(Q)$ and $N_G[Q]$ to denote the sets $N_G(V(Q))$ and $N_G[V(Q)]$, respectively. For a collection of paths/cycles $\mathcal{Q}$, we use $|\mathcal{Q}|$ to denote the number of paths/cycles in $\mathcal{Q}$ and $V(\mathcal{Q})$ to denote the set $\bigcup_{Q \in \mathcal{Q}} V(Q)$. We say a path P in G is a *degree-two path* if all vertices in $V(P)$, including the end vertices of P, have degree exactly 2 in G. We say P is a *maximal degree-two path* if no proper superset of P also forms a degree-two path. We note that the notions of *walks* and *closed walks* are defined exactly as paths and cycles, respectively, except that their vertices need not be distinct. Finally, a *feedback vertex set* is a subset F of vertices such that $G-F$ is a forest.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

3. Basic resullts.

Below we formally state and prove some of the key results that will be used throughout the paper, starting with the classic Erdős–Pósa theorem [16]. In particular, we give an algorithmic version of the Erdős–Pósa theorem that runs in time linear in $|V|$ and which outputs either a solution or a small feedback vertex set (Theorem 3.2).

PROPOSITION 3.1 (see [16]). *There exists a constant c' such that every (multi) graph either contains k vertex-disjoint cycles or has a feedback vertex set of size at most $c'k\log k$.*

Observe that any (multi) graph $G = (V, E)$ whose feedback vertex set number is bounded by $c'k\log k$ has less than $5c'k\log k \cdot |V|$ edges. (Recall that we restrict the multiplicity of an edge to be 2.) Indeed, letting F denote a feedback vertex set of minimum size, the worst case (in terms of $|E|$) is obtained when $G-F$ is a tree, which contains $|V|-|F|-1$ edges, and between every pair of vertices $v \in F$ and $u \in V$, there exists an edge of multiplicity 2, and between every pair of (not necessarily distinct) vertices in F, there also exists an edge of multiplcity 2. Thus, by Proposition 3.1, in the case $|E| > |V| - |F| - 1 + 2|F||V| + 2|F|^2 \geq 5c'k\log k \cdot |V|$, the input instance, (G, k) of CYCLE PACKING, is a yes-instance, and after we discard an arbitrary set of $|E|-5c'k\log k\cdot|V|$ edges, it remains a yes-instance. A simple operation which discards at least $|E| - 5c'k\log k \cdot |V|$ edges and can be performed in time $\mathcal{O}(k\log k \cdot |V|)$ is described after the assumption below.

Assumption 1. We assume that $|E| = \mathcal{O}(k\log k \cdot |V|)$.

Before we proceed, given a graph $G = (V, E)$, let us formally argue how to discard at least $|E| - 5c'k\log k \cdot |V|$ edges in time $\mathcal{O}(k\log k \cdot |V|)$. We examine the vertices in V in some arbitrary order $\{v_1, v_2, \dots, v_{|V|}\}$ and initialize a counter x to 0. For each vertex v_i, if $x < 5c'k\log k \cdot |V|$, then we iterate over the set of edges incident to v_i, and for each edge whose other endpoint is v_j for $j \geq i$, we increase x by 1. Let ℓ be the largest index for which we iterated over the set of edges incident to v_ℓ. We copy V and initialize the adjacency lists to be empty. Then, we copy the adjacency lists of the vertices $v_1, v_2, \dots, v_\ell$, where for each adjacency involving vertices v_i and v_j, where $i \leq \ell < j$, we update the adjacency list of v_j to include v_i. This completes the description of the procedure.

Now, we state our algorithmic version of Proposition 3.1. The proof partially builds upon the proof of the Erdő–Pósa theorem in the book [13].

THEOREM 3.2. *There exists a constant c and a polynomial-space algorithm such that given a (multi) graph G and a nonnegative integer k, in time $k^{\mathcal{O}(1)} \cdot |V|$ it either outputs k vertex-disjoint cycles or a feedback vertex set of size at most $ck\log k = r$.*

Proof. We fix c as the smallest integer such that $c \geq 150(\log_2 c)$. Let $G = (V, E)$ be a (multi) graph, and let k be a nonnegative integer. The objective is to show that in time $k^{\mathcal{O}(1)} \cdot |V|$ we can either output k vertex-disjoint cycles or a feedback vertex set of size at most $ck\log k = r$. We remark that the first part of this proof, which ends at the statement of Lemma 3.3, follows the proof of the Erdős–Pósa theorem [16] given in the book [13].

We may assume that G contains at least one cycle, since this fact can clearly be checked in time $\mathcal{O}(|V| + |E|)$, and if it is not true, we output an empty set as a feedback vertex set. Now, we construct a maximal subgraph H of G such each vertex in H is of degree 2 or 3 (in H). This construction can be done in time $\mathcal{O}(|V| + |E|)$ (see [2]). Let V_2 and V_3 be the degree-2 and degree-3 vertices in H, respectively. We also compute (in time $\mathcal{O}(|V| + |E|)$) the set $\mathcal{S}$ of connected components of $G - V(H)$.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Observe that for each connected component $S \in \mathcal{S}$, there is at most one vertex $v_S \in V_2$ such that there is at least one vertex in S adjacent to v_S, else we obtain a contradiction to the maximality of H as it could have been extended by adding a path from S. We compute (in time $\mathcal{O}(|V| + |E|)$) the vertices v_S, where for each component for which v_S is undefined (since it does not exist), we set $v_S = nil$. Let $V_2^\star \subseteq V_2$ be the set of vertices $v_S \neq nil$ such that v_S has at least two neighbors in S, which is easily found in time $\mathcal{O}(|V| + |E|)$. Observe that if $|V_2^\star| \geq k$, we can output k vertex-disjoint cycles in time $\mathcal{O}(|V| + |E|)$. Thus, we next assume that $|V_2^\star| < k$. Moreover, observe that $V_2^\star \cup V_3$ is a feedback vertex set. Thus, if $|V_2^\star \cup V_3| \leq ck \log k$, we are done. We next assume that $|V_2^\star \cup V_3| > ck \log k$. In particular, it holds that $|V_3| > ck \log k - k \geq (c-1)k \log k$.

Let H^* be the graph obtained from H by contracting, for each vertex in V_2, an edge incident to it. We remark that here we permit the multiplicity of edges to be 3. Then, H^* is a cubic graph whose vertex-set is V_3. To find k vertex-disjoint cycles in G in time $k^{\mathcal{O}(1)} \cdot |V|$, it is sufficient to find k vertex-disjoint cycles in H^* in time $k^{\mathcal{O}(1)} \cdot |V|$, since the cycles in H^* can be translated into cycles in G in time $\mathcal{O}(|V| + |E|)$. We need to rely on the following claim, whose proof is given in the book [13]. We remark that the original claim refers to graphs, but it also holds for multigraphs.

PROPOSITION 3.3 (see [13, Theorem 2.3.2, p. 13]). *If a cubic (multi) graph contains at least $q = 4k(\log k + \log \log k + 4)$ vertices, then it contains k vertex-disjoint cycles.*

Thus, we know that H^* contains k vertex-disjoint cycles, and it remains to find them in time $k^{\mathcal{O}(1)} \cdot |V|$. We now modify H^* to obtain a cubic graph H' on at least q vertices but at most $\mathcal{O}(k \cdot \log k)$ vertices, such that given k vertex-disjoint cycles in H', we can translate them into k vertex-disjoint cycles in H^* in time $\mathcal{O}(|V|)$, which will complete the proof. To this end, we initially let H' be a copy of H^*. Now, as long as $|V(H')| > (c-1)k \log k + 2$, we perform the following procedure:

1. Choose arbitrarily a vertex $v \in V(H')$.
2. If v has exactly one neighbor u—that is, $\{v, u\}$ is an edge of multiplicity 3—remove v and u from the graph.
3. Else if v has a neighbor u such that u, in turn, has a neighbor w (which might be v) such that the edge $\{u, w\}$ is of multiplicity 2, then remove u and w from H' and connect the remaining neighbor of u to the remaining neighbor of w by a new edge (which might be a self-loop).
4. Else, let x, y, z be the three distinct neighbors of v. Then, remove v and add an edge between x and y. Now, each vertex is of degree 3, except for z, which is of degree 2, and has two distinct neighbors. Remove z, and connect its two neighbors by an edge.

Since this procedure runs in time $\mathcal{O}(1)$ and each call decreases the number of vertices in the graph, the entire process runs in time $\mathcal{O}(|V|)$. It is also clear that the procedure outputs a cubic graph, and at its end, $(c-1)k \log k \leq |V(H')| \leq (c-1)k \log k + 2$. Thus, to prove the correctness of the process, it is now sufficient to consider graphs H_1 and H_2, where H_2 is obtained from H_1 by applying the procedure once, and show that given a set $\mathcal{C}_2$ of k vertex-disjoint cycles in H_2, we can modify them to obtain a set $\mathcal{C}_1$ of k vertex-disjoint cycles in H_1. Let v be the vertex chosen in the first step. If the condition in the second step was true, we simply let $\mathcal{C}_1 = \mathcal{C}_2$. In the second case, we examine whether the newly added edge belongs to a cycle in the solution in time $\mathcal{O}(1)$ (as we assume that each element in the graph, if it belongs to the solution, has a pointer to its location in the solution), and if it is true, we replace it by the path between its endpoints

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

whose only internal vertices are u and w. Finally, suppose the procedure reached the last case. Then, if the first newly added edge is used, replace it by the path between its endpoints, x and y, whose only internal vertex is v, and if the second newly added edge is used, replace it by the path between its endpoints whose only internal vertex is z.

We are now left with the task of finding k vertex-disjoint cycles in H'. We initialize a set $\mathcal{C}$ of vertex-disjoint cycles to be empty. As long as $|\mathcal{C}| < k$, we find a shortest cycle in H' in time $\mathcal{O}(|V(H')| \cdot |E(H')|) = k^{\mathcal{O}(1)}$ (see [24]), insert it into $\mathcal{C}$, and remove all of the edges incident to its vertices from H'. Thus, to conclude the proof, it remains to show that for each $i \in \{0, 1, \ldots, k-1\}$, after we remove the edges incident to the ith cycle from H', it still contains a cycle.

By using induction on i, we show that after removing the edges incident to the ith cycle from H', the number of edges in H' is at least $p(i) = \frac{3}{2}(c-1)k\log_2 k - 12 \cdot i \cdot \log_2(ck\log_2 k)$. This would imply that the average degree of a vertex of H' is at least $\frac{2p(i)}{|V(H')|} \geq \frac{2p(i)}{(c-1)k\log_2 k+2} \geq 2$ (we later also explicitly show that $2p(i) \geq (\sqrt{2}+1)ck\log_2 k$), and therefore it contains a cycle (since the average degree of a forest is smaller than 2). Initially, H' is a cubic graph, and therefore $|E(H')| = \frac{3}{2}|V(H')| \geq \frac{3}{2}(c-1)k\log_2 k$, and the claim is true. Now, suppose that it is true for some $i \in \{0, 1, \ldots, k-2\}$, and let us prove that it is true for $i+1$. By Proposition 3.4, a shortest cycle in H' is of length at most $2\log_{d-1}|V(H')|+2 \leq 3\log_{d-1}(ck\log_2 k)$, where $d = \frac{2p(i)}{(c-1)k\log_2 k+2} \geq \frac{2p(i)}{ck\log_2 k}$. Such a cycle is incident to at most $6\log_{d-1}(ck\log_2 k)$ edges. Therefore, after removing from H' the edges incident to a shortest cycle in it, it contains at least $p(i) - 6\log_{d-1}(ck\log_2 k) \geq p(i) - 6\frac{\log_2(ck\log_2 k)}{\log_2(d-1)} = p(i) - 6\frac{\log_2(ck\log_2 k)}{\log_2(\frac{2p(i)}{ck\log_2 k}-1)}$ edges. Thus, by the induction hypothesis, it remains to prove that $\log_2(\frac{2p(i)}{ck\log_2 k}-1) \geq 1/2$, to which end we need to show that $\frac{2p(i)}{ck\log_2 k} - 1 \geq \sqrt{2}$, that is, $2p(i) \geq (\sqrt{2}+1)ck\log_2 k$. For this purpose, it is sufficient to show that $4p(i) \geq 5ck\log_2 k$. By the induction hypothesis and since $i \leq k-1$, $4p(i) \geq 6(c-1)k\log_2 k - 48k\log_2(ck\log_2 k) = 5ck\log_2 k + (ck\log_2 k - 6k\log_2 k - 48k\log_2 k - 48k\log_2 c - 48k\log_2\log_2 k) \geq 5ck\log_2 k + (ck\log_2 k - 150(\log_2 c)k\log_2 k)$. Thus, we need to show that $c \geq 150(\log_2 c)$, which holds by our choice of c. This concludes the proof. ∎

Next, we state two results relating to cycles of average and short lengths.

PROPOSITION 3.4 (see [1]). *Any (multi) graph $G = (V, E)$ on n vertices with average degree d contains a cycle of length at most $2\log_{d-1} n + 2$.*

Itai and Rodeh [24] showed that given a (multi) graph $G = (V, E)$, an "almost" shortest cycle (if there is any) in G can be found in time $\mathcal{O}(|V|^2)$. To obtain a linear dependency on $|V|$ (given a small feedback vertex set), we prove the following result.

LEMMA 3.5. *Given a (multi) graph $G = (V, E)$ and a feedback vertex set F of G, a shortest cycle (if there is any) in G can be found in time $\mathcal{O}(|F| \cdot (|V| + |E|))$.*

Proof. We can clearly detect self-loops and edges of multiplicity 2 in time $\mathcal{O}(|V| + |E|)$, and return a cycle of length 1 or 2 accordingly, and therefore we next assume that G is a simple graph. Since F is a feedback vertex set, to prove the lemma it is sufficient to present a procedure that given a vertex $v \in F$, finds in time $\mathcal{O}(|V| + |E|)$ a cycle that is at least as short as the shortest cycle in G that contains v. Indeed, then we can iterate over F and invoke this procedure, returning the shortest cycle among those returned by the procedure. Thus, we next fix some vertex $v \in F$. Let H be the connected component of G containing v.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

From the vertex v, we run a breadth first search (BFS). Thus, we obtain a BFS tree T rooted at v, and each vertex in V gets a level i, indicating the distance between this vertex and v (the level of v is 0). By iterating over the neighborhood of each vertex, we identify the smallest index i_1 such that there exists an edge with both endpoints, u_1 and v_1, at level i_1 (if such an index exists), and the smallest index i_2 such that there exists a vertex w_2 at level i_2 adjacent to two vertices, u_2 and v_2, at level $i_2 - 1$ (if such an index exists). For i_1, the edge $\{u_1, v_1\}$ and the paths between v_1 and u_1 and their lowest common ancestor result in a cycle of length at most $2i_1+1$. For i_2, the edges $\{w_2, u_2\}$ and $\{w_2, v_2\}$ and the paths between u_2 and v_2 and their lowest common ancestor result in a cycle of length at most $2i_2$. We return the shorter cycle among the two (if such a cycle exists).

Suppose that there exists a cycle containing v, and let C be a shortest such cycle. We need to show that the above procedure returns a cycle at least as short as C. Every edge of H either connects two vertices of the same level, or a vertex of level $i-1$ with a vertex of level i. Thus, if there does not exist an index i'_1 such that there exists an edge in $E(C)$ with both endpoints, u'_1 and v'_1, at level i'_1, there must exist an index i'_2 such that there exists a vertex w'_2 at level i'_2 adjacent to two vertices, u'_2 and v'_2, at level $i'_2 - 1$, and the edges $\{w'_2, u'_2\}$ and $\{w'_2, v'_2\}$ belong to $E(C)$. First, suppose that the first case is true. Then, the procedure returns a cycle of length at most $2i'_1+1$. The length of C cannot be shorter than $2i'_1+1$, since it consists of a path from v to u'_1 (whose length is at least i'_1 since u'_1 belongs to level i'_1), a path from v to v'_1 whose only common vertex with the previous path is v (whose length is at least i'_1 since v'_1 belongs to level i'_1), and the edge $\{u'_1, v'_1\}$. Now, suppose that the second case is true. Then, the procedure returns a cycle of length at most $2i'_2$. The length of C cannot be shorter than $2i'_2$, since it consists of two internally vertex-disjoint paths from v to w'_2 (each of length at least i'_2 since w'_2 belongs to level i'_2). ∎

Finally, we state a result that will be used (in Lemma 4.3) to bound the size of a graph we obtain after performing simple preprocessing operations as well as repetitive removal of short cycles.

Proposition 3.6 (see [37, Lemma 9]). *Let $T = (V, E)$ be a forest on N vertices. Let $M' = \{\{i, j\} \in E \mid deg_T(i) = deg_T(j) = 2\}$ and $L = \{a \in V \mid deg_T(a) \leq 1\}$. Then there exists $M \subseteq M'$ such that M is a matching and $|W| \geq \frac{N}{4}$, where $W = L \cup M$.*

4. Removing leaves, induced paths, and short cycles.

In this section we give a structural decomposition of the graph, which is "essentially" obtained by repeatedly removing vertices of a short cycle from the graph. As is usually the case when dealing with cycles in a graph, we start by defining three rules which help in getting rid of vertices of degree at most 2 as well as edges of multiplicity larger than 2.[3] It is not hard to see that all three Reduction Rules A1, A2, and A3 are safe, i.e., they preserve solutions in the reduced graph.

Reduction Rule A1. *Delete vertices of degree at most 1.*

Reduction Rule A2. *If there is a vertex v of degree exactly 2 that is not incident to a self-loop, then delete v and connect its two (not necessarily distinct) neighbors by a new edge.*

Reduction Rule A3. *If there is a pair of vertices u and v in V such that $\{u, v\}$ is an edge of multiplicity larger than 2, then reduce the multiplicity of the edge to 2.*

[3]Recall that we have used the bound 2 of the mutiplicity of any edge to bound $|E|$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Observe that the entire process that applies these rules exhaustively can be done in time $\mathcal{O}(|V| + |E|) = \mathcal{O}(k \log k \cdot |V|)$. Indeed, in time $\mathcal{O}(|V|)$ we first remove the vertex-set of each maximal path between a leaf and a degree-two vertex. No subsequent application of Reduction Rule A2 or A3 creates vertices of degree at most one. Now, we iterate over the set of degree-two vertices. For each degree-two vertex that is not incident to a self-loop, we apply Reduction Rule A2. Next, we iterate over E, and for each edge of multiplicity larger than two, we apply Reduction Rule A3. At this point, the only new degree-two vertices that can be created are vertices incident to exactly one edge, whose multiplicity is two. Therefore, during one additional phase where we exhaustively apply Reduction Rule A2, the only edges of multiplicity larger than two that can be created are self-loops. Thus, after one additional iteration over E, we can ensure that no rule among Reduction Rules A1, A2, and A3 is applicable.

Since these rules will be applied iteratively, we define an operator, denoted by $\mathsf{reduce}(G)$, that takes as input a graph G and returns the (new) graph G' that results from exhaustive applications of Reduction Rules A1, A2, and A3.

Definition 4.1. *For a (multi) graph G, we let $G' = \mathsf{reduce}(G)$ denote the graph obtained after exhaustive applications of Reduction Rules* A1, A2, *and* A3. $|\mathsf{reduce}(G)|$ *denotes the number of vertices in* $\mathsf{reduce}(G)$. *Every vertex v in G is either deleted or belongs to* $\mathsf{reduce}(G)$*—for the sake of clarity, we treat the vertex v in* $\mathsf{reduce}(G)$ *as if it has its own identity and refer to it as the image of v. Accordingly, we let* $\mathsf{img}(\mathsf{reduce}(G))$ *denote the preimage of the vertex set of* $\mathsf{reduce}(G)$, *i.e.,* $\mathsf{img}(\mathsf{reduce}(G))$ *is the set of vertices in G which are not deleted in* $\mathsf{reduce}(G)$.[4]

Observation 1. *For a graph $G = (V, E)$ and a set $E' \subseteq \binom{V}{2}$ it holds that* $|\mathsf{reduce}(G + E')| \leq |\mathsf{reduce}(G)| + 2|E'|$.

The first step of our algorithm consists of finding, in time linear in $|V|$, a set S satisfying the conditions specified in Lemmas 4.2 and 4.3. Intuitively, S will contain vertices of "short" cycles in the input graph, where short will be defined later.

Lemma 4.2. *Given a (multi) graph $G = (V, E)$ and two integers $k > 0$ and $g > 6$, there exists a $k^{\mathcal{O}(1)} \cdot |V|$-time algorithm that either finds k vertex-disjoint cycles in G or finds a (possibly empty) set $S \subseteq V$ such that girth*$(\mathsf{reduce}(G-S)) > g$ *and* $|S| < gk$.

Proof. We proceed by constructing such an algorithm. First, we apply the algorithm of Theorem 3.2, which outputs either k vertex-disjoint cycles or a feedback vertex set F of size at most $ck \log k = r$. In the former case we are done. In the latter case, i.e., the case where a feedback vertex set F is obtained, we apply the following procedure iteratively (initially, we set $S = \emptyset$):

(1) Apply Lemma 3.5 to find a shortest cycle C in $\mathsf{reduce}(G)$.
(2) If no cycle was found or $|C| > g$, then return S.
(3) Otherwise, i.e., if $|C| \leq g$, then add the vertices of C to S, delete those vertices from G to obtain G', set $G = G'$, and repeat from step (1).

Note that if step (3) is applied k times, then we can terminate and return the corresponding k vertex-disjoint cycles in G. Hence, when the condition of step (2) is satisfied, i.e., when the described procedure terminates, the size of S is at most $g(k-1) < gk$ and girth$(\mathsf{reduce}(G - S)) > g$. Since the algorithm of Theorem 3.2 runs in time $k^{\mathcal{O}(1)} \cdot |V|$, and each iteration of steps (1)–(3) is performed in time $\mathcal{O}((k \log k)^2 \cdot |V|)$, we obtain the desired time complexity. ◻

[4]For example, if G is defined by $V(G) = \{a, b, c, d, e\}$ and $E(G) = \{\{a, b\}, \{a, b\}, \{b, c\}, \{c, d\}, \{c, d\}, \{d, e\}$, then $\mathsf{reduce}(G)$ is defined by the vertex set $\{b, c\}$ and edge set $\{\{b\}, \{b, c\}, \{c\}\}$, and $\mathsf{img}(\mathsf{reduce}(G)) = \{b, c\} \subseteq V(G)$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

LEMMA 4.3. *Given a (multi) graph $G = (V, E)$ and two integers $k > 0$ and $g > 6$, let S denote the set obtained after applying the algorithm of Lemma 4.2 (assuming no k vertex-disjoint cycles obtained). Then $|\mathsf{reduce}(G - S)| \leq (2ck \log k)^{1+\frac{6}{g-6}} + 3ck \log k$.*

Proof. Let $G' = (V', E') = \mathsf{reduce}(G - S)$ and $|V'| = n'$. First, recall that G admits a feedback vertex set of size at most $ck \log k = r$. Since Reduction Rules A1, A2, and A3 do not increase the feedback vertex set of the graph (see, e.g., [37, Lemma 1]), G' also admits a feedback vertex set F of size at most r. Let T denote the induced forest on the remaining $N = n' - r$ vertices in G'. Moreover, from Lemma 4.2, we know that $\mathrm{girth}(G') > g > 6$.

Next, we apply Proposition 3.6 to T to get W. Now with every element $a \in W$ we associate an unordered pair of vertices of F as follows. Assume $a \in L$, i.e., a is a vertex of degree 0 or 1. Since the degree of a is at least 3 in G', a has at least two neighbors in F. We pick two of these neighbors arbitrarily and associate them with a. We use $\{x_a, y_a\}$ to denote this pair. If $a = \{u, v\}$ is an edge from M, then each of u and v has degree at least 3 in G' and each has at least one neighbor in F. We pick one neighbor for each and associate the pair $\{x_u, x_v\}$ with a. Note that since $\mathrm{girth}(G') > 6$, $x_u \neq x_v$ and $x_a \neq y_a$.

We now construct a new multigraph $G^\star = (V^\star, E^\star)$ with vertex set $V^\star = F$ as follows. For every vertex $a \in W$ we include an edge in $E^\star$ between x_a and y_a, and for every edge $a = \{u, v\} \in W$ we include an edge in $E^\star$ between x_u and x_v. By Proposition 3.6, we know that W is of size at least $\frac{N}{4}$. It follows that $G^\star$ has at least $\frac{N}{4}$ edges and hence its average degree is at least $\frac{N}{2r}$ as $|V^\star| = ck \log k = r$. Note that if $G^\star$ has a cycle of length at most ℓ, then G' has a cycle of length at most 3ℓ, as any edge of the cycle in $G^\star$ can be replaced by a path of length at most 3 in G'. Combining this with the fact that $\mathrm{girth}(G') > g > 6$, we conclude that $G^\star$ contains no self-loops or parallel edges. Hence $G^\star$ is a simple graph with average degree at least $\frac{N}{2r}$. By Proposition 3.4, $G^\star$ must have a cycle of length at most

$$2 \log_{\frac{N}{2r}-1} r + 2 = \frac{2 \log r}{\log(\frac{N}{2r} - 1)} + 2,$$

which implies that G' must have a cycle of length at most

$$\frac{6 \log r}{\log(\frac{N}{2r} - 1)} + 6.$$

Finally, by using the fact that $\mathrm{girth}(G') > g$ and substituting N and r, we get

$$\begin{aligned}
\frac{6 \log r}{\log(\frac{N}{2r} - 1)} + 6 > g &\iff \log r > \frac{(g-6)}{6} \log\left(\frac{N - 2r}{2r}\right) \\
&\iff \log r > \frac{(g-6)}{6} \log(N - 2r) - \frac{(g-6)}{6} \log(2r) \\
&\iff \frac{\log r + \frac{(g-6)}{6} \log(2r)}{\frac{(g-6)}{6}} > \log(N - 2r) \\
&\implies \frac{\log(2r) + \frac{(g-6)}{6} \log(2r)}{\frac{(g-6)}{6}} > \log(N - 2r) \\
&\iff \left(1 + \frac{6}{g-6}\right) \log(2r) > \log(N - 2r)
\end{aligned}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

$$\iff \left(1+\frac{6}{g-6}\right)\log(2ck\log k) > \log(n' - 3ck\log k)$$

$$\iff (2ck\log k)^{1+\frac{6}{(g-6)}} + 3ck\log k > n'.$$

This completes the proof. ∎

The usefulness of Lemma 4.3 comes from the fact that by setting $g = \frac{48\log k}{\log\log k} + 6$, we can guarantee that $|\,\mathsf{reduce}(G-S)| < 3ck\log k + 2ck\log^{1.5} k$, and therefore we can beat the $\mathcal{O}(k\log^2 k)$ bound. That is, we have the following consequence.

COROLLARY 4.4. *Given a (multi) graph $G = (V, E)$ and an integer $k > 0$, let S denote the set obtained after applying the algorithm of Lemma* 4.2 *with* $g = \frac{48\log k}{\log\log k} + 6$ *(assuming no k vertex-disjoint cycles obtained). Then* $|\,\mathsf{reduce}(G-S)| \le 3ck\log k + 2ck\log^{1.5} k$.

Proof. By Lemma 4.3, $|\,\mathsf{reduce}(G-S)| \le (2ck\log k)^{1+\frac{\log\log k}{8\log k}} + 3ck\log k$. Assuming $k > \log k > c > 2$, we have $(2ck\log k)^{1+\frac{\log\log k}{8\log k}} = (2ck\log k)(2ck\log k)^{\frac{\log\log k}{8\log k}} \le (2ck\log k)k^{\frac{4\log\log k}{8\log k}}$. Now note that $k^{\frac{4\log\log k}{8\log k}} \le \log^{0.5} k$. Hence, $(2ck\log k)^{1+\frac{\log\log k}{8\log k}} \le 2ck\log k\log^{\frac{1}{2}} k \le 2ck\log^{1.5} k$. This completes the proof. ∎

5. Bounding the core of the remaining graph.

At this point, we assume, without loss of generality, that we are given a graph $G = (V, E)$, a positive integer k, $g = \frac{48\log k}{\log\log k} + 6$, and a set $S \subseteq V$ such that girth$(\mathsf{reduce}(G-S)) > g$, $|S| < gk$, and $|\,\mathsf{reduce}(G-S)| \le 3ck\log k + 2ck\log^{1.5} k$.

Even though the number of vertices in $\mathsf{reduce}(G-S)$ is bounded, the number of vertices in $G-S$ is unbounded. In what follows, we show how to bound the number of "objects" in $G-S$, where an object is either a vertex in $G-S$ or a degree-two path in $G-S$. The next lemma is an extension of a lemma by Lokshtanov et al. [34, Lemma 5.2]. Specifically, the lemma by Lokshtanov et al. [34] does not output the sets V_X and E_X but only one vertex or one edge,[5] and furthermore, the running time is only analyzed to have polynomial dependency on the input size. We need to have a linear dependency on $|V|$, and furthermore that the entire algorithm will eventually have linear dependency on $|V|$. In particular, the latter requirement is the reason why we need to output sets V_X and E_X rather than only one vertex or one edge, else the usage of the lemma will entail a quadratic dependency on $|V|$.

LEMMA 5.1. *Let $G = (V, E)$ be a (multi) graph and let $X \subseteq V$ be any subset of the vertices of G. Suppose there are more than $|X|^2(2|X|+1)$ vertices in $G-X$ whose degree in $G-X$ is at most one. Then, there is either an isolated vertex w in $G-X$ or an edge $e \in E$ such that (G, k) is a yes-instance of* CYCLE PACKING *if and only if either $(G-\{w\}, k)$ or $(G/e, k)$ is a yes-instance. Moreover, there is an $\mathcal{O}(|X|^2 \cdot k\log k \cdot |V|)$-time algorithm that given G and X, outputs sets $V_X \subseteq V \setminus X$ and $E_X \subseteq E(G-X)$ such that, for the graph $G' = (G/E_X) - V_X$, it holds that (G, k) is a yes-instance of* CYCLE PACKING *if and only if (G', k) is a yes-instance of* CYCLE PACKING, *and $G'-X$ contains at most $|X|^2(2|X|+1)$ vertices whose degree in $G'-X$ is at most one.*

Proof. In $G-X$ each $x \in L(u, v)$ is adjacent to both u and v (if $u = v$, then $L(u, u)$ is the set of vertices which have degree at most one in $G-X$ and an edge of multiplicity two to u). For each pair $(u, v) \in X \times X$, we arbitrarily mark $2|X|+1$

[5] Additionally, the lemma by Lokshtanov et al. [34] assumes that X is a feedback vertex set.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

vertices from $L(u,v)$ if $|L(u,v)| > 2|X|+1$, and we mark all vertices in $L(u,v)$ if $|L(u,v)| \leq 2|X|+1$. We can execute this process as follows. First, in time $\mathcal{O}(|X|\cdot|V|)$, for each vertex in X we compute the set of its neighbors of degree at most one in $G-X$. Then, in time $\mathcal{O}(|X|^3)$, for each pair $(u,v) \in X \times X$ we mark at most $2|X|+1$ vertices as required.

Since we mark at most $2|X|+1$ vertices for each pair $(u,v) \in X \times X$, there can be at most $|X|^2(2|X|+1)$ marked vertices in $G-X$. Let w be an unmarked vertex of degree at most one in $G-X$. We only consider the case where $\deg_{G-X}(w) = 1$, as the other case can be proved analogously. Let e be the unique edge in $G-X$ which is incident to w and let z be the other endpoint of this edge. Let $\mathcal{C}$ be a set of maximum size of vertex-disjoint cycles in G. Observe that if $\mathcal{C}$ does not contain a pair of cycles such that each of them intersects a different endpoint of e, then contracting e keeps the resulting cycles vertex disjoint in G/e. Therefore, we may assume that $\mathcal{C}$ contains two cycles C_w and C_z, where C_w contains w and C_z contains z. The neighbor(s) of w in C_w must lie in X. Let these neighbors be x and y. (Again, x and y are not necessarily distinct.) Since $w \in L(x,y)$ and it is unmarked, there are $2|X|+1$ other vertices in $L(x,y)$ which were marked by the marking procedure. Moreover, each degree-1 vertex in $G-X$ that belongs to a cycle in $\mathcal{C}$ is either the predecessor or the successor of a vertex in X in such a cycle. Therefore, at most $2|X|$ of the marked vertices can participate in cycles in $\mathcal{C}$. Hence, there exists a vertex in $L(x,y)$, call it w', which is unused by $\mathcal{C}$. Consequently, we can route the cycle C_w through w' instead of w, which gives us a set of $|\mathcal{C}|$ vertex disjoint cycles in G/e.

The first phase of the claimed $\mathcal{O}(|X|^2 \cdot k\log k \cdot |V|)$-time algorithm performs the above marking procedure, and then proceeds as follows. First it deletes every unmarked isolated vertex in $G-X$. Then, it contracts every edge in $G-X$ incident to at least one unmarked vertex of degree one in $G-X$. After these operations, new vertices in $G-X$ of degree at most one in $G-X$ might have been created. These vertices were either the unique neighbors in $G-X$ of deleted vertices or vertices incident to contracted edges. Thus, in the case new vertices in $G-X$ of degree at most one in $G-X$ have been created, the algorithm performs another phase. Here, the algorithm iterates over the set of new vertices in $G-X$ of degree at most one in $G-X$, and for each such vertex, if it is a neighbor of two vertices in X for which we have not yet marked $2|X|+1$ vertices, the algorithm marks it. Then, the algorithm deletes vertices and contracts edges as it did in the first phase. The running time of such a phase is bounded by $\mathcal{O}(|X|^2\cdot\rho)$, where ρ is the total number of vertices deleted and edges contracted in the previous phase. As long as new degree-one vertices are created, the execution of the algorithm continues. Since each vertex can be deleted only once, and each edge can be contracted only once, the overall running time is bounded by $\mathcal{O}(|X|(|X|^2+|V|)+|X|^2\cdot(|V|+|E|)) = \mathcal{O}(|X|^2\cdot k\log k\cdot|V|)$ (since $|E| = \mathcal{O}(k\log k\cdot|V|)$). It also holds that when the algorithm terminates, $G-X$ contains at most $|X|^2(2|X|+1)$ vertices whose degree in $G-X$ is at most one. This completes the proof of the lemma. ∎

Armed with Lemma 5.1, we are now ready to prove the following result. For a forest T, we let $T_{\leq 1}$, T_2, and $T_{\geq 3}$ denote the sets of vertices in T having degree at most one in T, degree exactly two in T, and degree larger than two in T, respectively. Moreover, we let $\mathcal{P}$ denote the set of all maximal degree-two paths in T.

LEMMA 5.2. *Let $G = (V,E)$, S, k, and g be as defined above. Let $R = \mathsf{img}(\mathsf{reduce}(G-S)) \subseteq (V \setminus S)$ denote the preimage of* $\mathsf{reduce}(G-S)$ *in $G-S$. Then, $T = G-S-R$ is a forest and for every maximal degree-2 path in $\mathcal{P}$ there are at most*

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

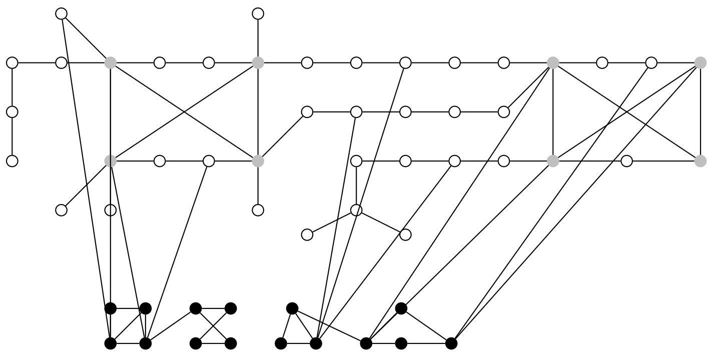

FIG. 1. *A graph G (not all edges shown), the set S (in black), the set R (in gray), and the set $T = G - R - S$ (in white).*

two vertices on the path having neighbors in R (in the graph $G - S$). Moreover, in time $k^{\mathcal{O}(1)} \cdot |V|$, we can guarantee that $|T_{\leq 1}|$, $|\mathcal{P}|$, and $|T_{\geq 3}|$ are bounded by $k^{\mathcal{O}(1)}$.

Proof. To see why $T = G - S - R$ must be a forest it is sufficient to note that for any cycle in $G - S$ at least one vertex from that cycle must be in $R = \mathsf{img}(\mathsf{reduce}(G - S))$ (see Figure 1). Recall that, since girth$(\mathsf{reduce}(G - S)) > 6$, every vertex in R has degree at least 3 in $G - S$. Now assume there exists some path $P \in \mathcal{P}$ having exactly three (the same argument holds for any number) distinct vertices u, v, and w (in that order) each having at least one neighbor in R (possibly the same neighbor). We show that the middle vertex v must have been in R, contradicting the fact that $T = G - S - R$. Consider the graph $G - S$ and apply Reduction Rules A1, A2, and A3 exhaustively (in $G - S$) on all vertices in the tree containing P except for u, v, and w. Regardless of the order in which we apply the reduction rules, the path P will eventually reduce to a path on three vertices, namely, u, v, and w. To see why v must be in R observe that even if the other two vertices have degree two in the resulting graph, after reducing them, v will have degree at least three (into R) and is therefore nonreducible.

Next, we bound the size of $T_{\leq 1}$, which implies a bound on the sizes of $T_{\geq 3}$ and $\mathcal{P}$. To do so, we simply invoke Lemma 5.1 by setting $X = S \cup R$. Since $|S| < gk$, $g = \frac{48 \log k}{\log \log k} + 6$ and $|R| \leq 3ck \log k + 2ck \log^{1.5} k$, we get that $|T_{\leq 1}| \leq |S \cup R|^2(2|S \cup R| + 1) = k^{\mathcal{O}(1)}$. Since in a forest, it holds that $|T_{\geq 3}| < |T_{\leq 1}|$, the bound on $|T_{\geq 3}|$ follows. Moreover, in a forest, it also holds that $|\mathcal{P}| < |T_{\leq 1}| + |T_{\geq 3}|$—if we arbitrarily root each tree in the forest at a leaf, one end vertex of a path in $\mathcal{P}$ will be a parent of a different vertex from $T_{\leq 1} \cup T_{\geq 3}$—and the bound on $|\mathcal{P}|$ follows as well. ∎

6. Guessing permutations.

In this section, given an instance (G, k) of CYCLE PACKING, we obtain $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ instances of CYCLE PACKING of the form (G', k), such that the size of G' is small and (G, k) is a yes-instance if and only if one of the output instances (G', k) is.

This section is devoted to proving the following lemma. Note that assuming the statement of the lemma, the only remaining task (to prove Theorem 1.1) is to develop an algorithm running in time $\mathcal{O}(2^{|V|} \cdot \text{poly}(|V|))$ and using polynomial space, which we present in section 7.

LEMMA 6.1. *Given an instance (G, k) of* CYCLE PACKING, *we can compute $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ instances of* CYCLE PACKING *of the form (G', k), where the number of*

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

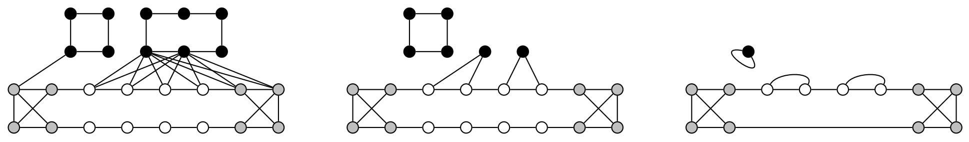

FIG. 2. *(Left) a graph G (not all edges shown), the set S (in black), the set R (in gray), and the set $T = G - R - S$ (in white). (Center) the graph obtained after guessing vertices in S and their neighbors in a solution. (Right) example of a reduced instance.*

vertices in G' is bounded by $\mathcal{O}(k \log^{1.5} k)$, such that (i) *$(G, k)$ is a yes-instance if and only if at least one of the instances (G', k) is a yes-instance and* (ii) *the instances are outputted one-by-one, so that the space complexity is polynomial and the* total *running time is $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$.*

Proof. We fix $g = \frac{48 \log k}{\log \log k} + 6$. Using Lemma 4.2, we first compute in time $k^{\mathcal{O}(1)} \cdot |V|$ a subset $S' \subseteq V$ such that $\text{girth}(\mathsf{reduce}(G - S')) > g$ and $|S'| < gk = \mathcal{O}(\frac{k \log k}{\log \log k})$. Then, we guess which vertices to delete from S' — that is, which vertices do not participate in a solution—in time $\mathcal{O}(2^{|S'|}) = 2^{\mathcal{O}(\frac{k \log k}{\log \log k})}$. Here, guesses refer to different choices which lead to the construction of different instances of CYCLE PACKING that are returned at the end—recall that we are allowed to return up to $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ different instances. Concretely, here we simply have a loop that iterates over all subsets of S', where each subset is a "guess" of which vertices of S' to delete. We now have a set $S \subseteq S'$ (of the vertices in S' that should not be deleted) such that $|S| = \mathcal{O}(\frac{k \log k}{\log \log k})$. By Corollary 4.4, we also know that $|\mathsf{reduce}(G-S)| = \mathcal{O}(k \log^{1.5} k)$.

Applying Lemma 5.2 with $R = \mathsf{img}(\mathsf{reduce}(G - S)) \subseteq (V \setminus S)$, we get a forest $T = G - (S \cup R)$ such that for every maximal degree-two path in $\mathcal{P}$ there are at most two vertices on the path having neighbors in R (in the graph $G - S$). In addition, the size of R is equal to $|\mathsf{reduce}(G - S)|$ which is bounded by $\mathcal{O}(k \log^{1.5} k)$, and the sizes $|T_{\leq 1}|$, $|\mathcal{P}|$ and $|T_{\geq 3}|$ are bounded by $k^{\mathcal{O}(1)}$ (see Figure 1).

For every vertex in S (which is assumed to participate in a solution), we now guess its two neighbors in a solution (see Figure 2). Note, however, that we only have a (polynomial in k) bound for $|S|$, $|R|$, $|T_{\leq 1}|$, $|\mathcal{P}|$, and $|T_{\geq 3}|$, but not for the length of paths in $\mathcal{P}$ and therefore not for the entire graph G. We let $Z_{\mathcal{P}}$ denote the set of vertices in $V(\mathcal{P})$ having neighbors in R. The size of $Z_{\mathcal{P}}$ is at most $2|\mathcal{P}|$. Moreover, we let $\mathcal{P}^\star$ denote the set of paths obtained after deleting $Z_{\mathcal{P}}$ from $\mathcal{P}$. Note that the size of $\mathcal{P}^\star$ is upper bounded by $|\mathcal{P}| + |Z_{\mathcal{P}}| \leq 3|\mathcal{P}|$ and that vertices in $V(\mathcal{P}^\star)$ are adjacent only to vertices in $V(\mathcal{P}^\star) \cup Z_{\mathcal{P}} \cup S$. Now, we create a set of "objects," $O = S \cup R \cup T_{\leq 1} \cup T_{\geq 3} \cup Z_{\mathcal{P}} \cup \mathcal{P}^\star$. We also let $\widetilde{O} = O \setminus \mathcal{P}^\star$. We then guess, for each vertex v in S, which two objects in O constitute its neighbors, denoted by $\ell(v)$ and $r(v)$, in a solution. Concretely, we have a loop that iterates over every mapping of each vertex in S to two objects in O. It is possible that $\ell(v) = r(v)$. Since $|O| = k^{\mathcal{O}(1)}$, we can perform these guesses in $k^{\mathcal{O}(\frac{k \log k}{\log \log k})}$, or equivalently $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$, time. We next assume that, for every vertex $v \in S$, if $\ell(v) \in \widetilde{O}$, then $\ell(v)$ is a neighbor of v in a solution, and otherwise (i.e., $\ell(v) \in \mathcal{P}$) v has a neighbor on the path $\ell(v)$. Indeed, if this assumption does not hold, then the current guess is not correct for a solution and hence we need not argue that in the current guess we find a solution subject to it—we only need to argue that for a correct guess (which we do make if there is a solution). The same claim holds for $r(v)$. Furthermore, we similarly assume that

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

if $\ell(v) = r(v) \in \widetilde{O}$, then $\{v, \ell(v)\}$ is an edge of multiplicity two, and otherwise if $\ell(v) = r(v)$, then v has (at least) two edges incident to vertices on the path $\ell(v)$.

Next, we fix some arbitrary order on $\mathcal{P}^\star$, and for each path in $\mathcal{P}^\star$, we fix some arbitrary orientation from one endpoint to the other. We let $S^\star$ denote the multiset containing two occurrences of every vertex $v \in S$, denoted by v_ℓ and v_r. We guess an order of the vertices in $S^\star$. The time spent for guessing such an ordering is bounded by $(2|S|)!$, which in turn is bounded by $2^{\mathcal{O}(\frac{k \log^2 k}{\log\log k})}$. The ordering, assuming it is guessed correctly, satisfies the following conditions. For each path $P \in \mathcal{P}^\star$, we let $\ell(P)$ and $r(P)$ denote the sets of vertices $v \in S$ such that $\ell(v) \in V(P)$ and $r(v) \in V(P)$, respectively. Now, for any two vertices $u, v \in \ell(P)$, if $u_\ell < v_\ell$ according to the order that we guessed, then the neighbor $\ell(u)$ of u appears before the neighbor $\ell(v)$ of v on P. Similarly, for any two vertices $u, v \in r(P)$, if $u_r < v_r$, then $r(u)$ appears before $r(v)$ on P. Finally, for any two vertices $u \in \ell(P)$ and $v \in r(P)$, if $u_\ell < v_r$, then $\ell(u)$ appears before $r(v)$ on P, and otherwise $r(v)$ appears before $\ell(u)$ on P.

Given a correct guess of $\ell(v)$ and $r(v)$ for each v in S, as well as a correct guess of a permutation of $S^\star$ for each path in $\mathcal{P}^\star$, we proceed as follows. We compute the set $\{x_v, y_v\}$ of two neighbors (now being vertices rather than objects) of each vertex $v \in S$ in a solution subject to our guesses. First, for each $v \in S$, if $\ell(v)$ $(r(v))$ is in $\widetilde{O}$, then $x_v = \ell(v)$ $(y_v = r(v))$. The remaining neighbors are assigned by a greedy procedure which agrees with the guessed permutation on $S^\star$; that is, for every path $P \in \mathcal{P}^\star$, we iterate over $\{v_\ell \mid v \in \ell(P)\} \cup \{v_r \mid v \in r(P)\}$ according to the guessed order, and to each vertex $v_\diamond$ in this set (where $\diamond \in \{\ell, r\}$), assign to v (as x_v or y_v depending on whether $\diamond$ is ℓ or r) the first neighbor of v on P that is after the last vertex on P that has already been assigned. (If such a vertex does not exist, we determine that the current guess is incorrect and proceed to the next one.) We let E_S be the set of edges incident on a vertex in S, and we let $E' = \{\{v, x_v\} \mid v \in S\} \cup \{\{v, y_v\} \mid v \in S\}$ denote the set of all guesses. In order to avoid introducing additional double edges, it is important to note that we do not add duplicates to E'. In other words, we only add a double edge between u and v when we have guessed that $x_v = y_v = u$ and $x_u = y_u = v$. Finally, to obtain an instance (G', k), we delete the edge set E_S from G, we add instead the set of edges E', and finally we apply the reduce operator, i.e., $G' = \mathsf{reduce}((G - E_S) + E')$.

CLAIM 1. *Let (G', k) be one of the instances generated by the above procedure. Then, the number of vertices in G' is bounded by $\mathcal{O}(k \log^{1.5} k)$.*

Proof. Recalling that by Corollary 4.4, we know that $|\mathsf{reduce}(G-S)| = \mathcal{O}(k \log^{1.5} k)$. Moreover, we have $|E'| = \mathcal{O}(|S|) = \mathcal{O}(\frac{k \log k}{\log\log k})$. Combining Observation 1 with the fact that $G' = \mathsf{reduce}((G - E_S) + E')$, we get $|\mathsf{reduce}((G - E_S) + E')| \leq |\mathsf{reduce}(G - E_S)| + 2|E'|$. Since in $G - E_S$ all vertices of S have degree zero, $|\mathsf{reduce}(G - E_S)| \leq |\mathsf{reduce}(G - S)|$. Hence, we conclude that $|\mathsf{reduce}((G - E_S) + E')| = \mathcal{O}(k \log^{1.5} k)$, as needed. ∎

CLAIM 2. *Let $G = (V, E)$, let k be a positive integer, let $g = \frac{48 \log k}{\log\log k} + 6$, and let S be a subset of V such that girth$(\mathsf{reduce}(G - S)) > g$, and $|S| < gk$. Then, (G, k) is a yes-instance if and only if at least one of the generated instances (G', k) is a yes-instance.*

Proof. First, given the safeness of our reduction rules, we assume that $G' = (G - E_S) + E'$ (and not $G' = \mathsf{reduce}((G - E_S) + E')$). This assumption is justified by the fact that any cycle packing in $(G - E_S) + E'$ can be easily recovered in $\mathsf{reduce}((G - E_S) + E')$, and vice versa. We assume that (G, k) is a yes-instance and we let $\mathcal{C} =$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

$\{C_1, C_2, \dots, C_k\}$ be a packing of k cycles, i.e., a set of k vertex-disjoint cycles in G. We let E_{cross} denote the set of edges connecting a vertex in S to a vertex in $\mathcal{P}^\star$. We call such edges cross edges. Recall that E' denotes the set of guessed edges during the construction of G'.

To prove the forward direction, we proceed by contradiction. Given (G, k) (assumed to be a yes-instance), we choose a solution $\mathcal{C}$ and a generated instance (G', k) as follows:

1. G' and $\mathcal{C}$ are in agreement with respect to our guessed permutation of $S^\star$.
2. All edges in $\mathcal{C}$ which are in E' but not in E_{cross} (chosen via brute-force) are retained in G'. Equivalently, $|(E' \setminus E_{\text{cross}}) \cap E(\mathcal{C})| = |(E' \setminus E_{\text{cross}})|$.
3. The number of edges (chosen greedily) in E' which are also contained in $\mathcal{C}$ is maximized. Equivalently, $|E' \cap E(\mathcal{C})|$ is maximized.

The fact that such a pair $(\mathcal{C}, (G', k))$ exists follows from the fact that for edges that are not in E_{cross} (possibly chosen greedily) we try all possibilities in a brute-force manner (no greedy choices are made). If $|E' \cap E(\mathcal{C})| = |E'|$, then we are done, as all the remaining edges in $\mathcal{C}$ are also contained in G'; therefore (G', k) is a yes-instance. If $|E' \cap E(\mathcal{C})| = q < |E'|$, then there exists at least one edge $e \in E' \setminus E(\mathcal{C})$. As previously noted, $e \in E_{\text{cross}}$. We let $e = \{u, v\}$, where $u \in S$, $v \in P$, $P \in \mathcal{P}^\star$, and there exists no edge $e' = \{u', v'\} \in E' \setminus E(\mathcal{C})$ satisfying all the aforementioned properties with v' occurring before v on P with respect to our fixed orientation. To complete the forward direction, we show how to modify $\mathcal{C}$ to obtain $\mathcal{C}'$ such that $|E' \cap E(\mathcal{C}')| = q + 1$, contradicting our initial assumption.

Since $u \in S$, it follows that $u \in V(\mathcal{C})$ (by assumption). Therefore, there must exist an edge $\{u, w\} \in E(\mathcal{C})$, where $w \neq v$ and w appears either before or after v on P. Assume that w appears after v on P. Let $\{v, z_0, z_1, \dots, w\}$ denote the subpath of P connecting v to w. Then, as all vertices in P have degree two in $G - S$, $\{z_0, z_1, \dots\} \cap V(\mathcal{C}) = \emptyset$, and assuming our guessed permutation is consistent with $\mathcal{C}$, if we replace $\{u, w\}$ by $\{u, v\}$ and $\{v, z_0, z_1, \dots, w\}$, we obtain a packing of k cycles $\mathcal{C}'$ in G', as needed. The case where w appears before v on P can be handled analogously.

For the other direction, let (G', k) be a yes-instance and let $\mathcal{C}' = \{C'_1, C'_2, \dots, C'_k\}$ be a cycle packing in G'. We again assume, without loss of generality, that $\mathcal{C}'$ is a cycle packing in $(G - E_S) + E'$, as one can trace back all reduction rules to obtain the graph $(G - E_S) + E'$. Since $(G - E_S) + E'$ is a subgraph of G, it follows that $\mathcal{C}'$ is also a packing of k cycles in G, completing the proof. ∎

Combining Claims 1 and 2 concludes the proof of the theorem. ∎

7. Dynamic programming and inclusion-exclusion.

Finally, we give an exact exponential-time algorithm for CYCLE PACKING. For this purpose, we use DP and the principle of inclusion-exclusion, inspired by the work of Nederlof [35].[6]

LEMMA 7.1. *There exists a (deterministic) polynomial-space algorithm that in time $\mathcal{O}(2^{|V|} \cdot \text{poly}(|V|))$ solves* CYCLE PACKING. *In the case a solution exists, it also outputs a solution.*

Proof. First, we recall the principle of inclusion-exclusion.

PROPOSITION 7.2 (folklore, [35]). *Let U and R be sets, and for every $v \in R$ let P_v be a subset of U. Use $\bar{P}_v$ to denote $U \setminus P_v$. With the convention $\bigcap_{v \in \emptyset} \bar{P}_v = U$, the*

[6]We remark that this result can also be proved by using the algorithm for the COVER POLYNOMIAL problem given in [35]; for the sake of being self-contained, we present the full details.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

following holds:

$$\left|\bigcap_{v\in R} P_v\right| = \sum_{F\subseteq R}(-1)^{|F|}\left|\bigcap_{v\in F}\bar{P}_v\right|.$$

We now proceed with the proof of Lemma 7.1. In the context of Proposition 7.2, define the universe U as the set of all tuples $(C_1,\ldots,C_k,\ w_1^1,\ldots,w_k^1,w_1^2,\ldots,w_k^2,L)$ such that each C_i is a closed walk in G of length at least three, w_i^1 and w_i^2 are consecutive occurrences of vertices in C_i, $L\subseteq V$, and $(\sum_{i=1}^k|V(C_i)|)+|L|=|V|$. Here, by $|V(C_i)|$ we refer to a multiset—that is, if C_i contains x occurrences of some vertex v, then $V(C_i)$ contains x occurrences of v as well. We define the requirement space $R=V$, and for each $v\in V$, we let P_v be the set of all tuples $(C_1,\ldots,C_k,w_1^1,\ldots,w_k^1,w_1^2,\ldots,w_k^2,L)\in U$ such that $v\in(\bigcup_{i=1}^k V(C_i))\cup L$. On the one hand, if G contains k vertex-disjoint cycles $C_1,\ldots,C_k$, then for any choice of edges $\{w_1^1,w_1^2\}\in E(C_1),\ldots,\{w_k^1,w_k^2\}\in E(C_k)$, we have

$$\left(C_1,\ldots,C_k,w_1^1,\ldots,w_k^1,w_1^2,\ldots,w_k^2,V\setminus\left(\bigcup_{i=1}^k V(C_i)\right)\right)\in\bigcap_{v\in V}P_v.$$

On the other hand, if there exists $(C_1,\ldots,C_k,w_1^1,\ldots,w_k^1\ ,w_1^2,\ldots,w_k^2,L)\in\bigcap_{v\in V}P_v$, then since $(\sum_{i=1}^k|V(C_i)|)+|L|=|V|$, each vertex $v\in V$ occurs exactly once in either exactly one of the closed walks C_i or in the set L. In this case, we conclude that $C_1,\ldots,C_k$ are vertex-disjoint cycles. Therefore, we need to accept the input instance if and only if $|\bigcap_{v\in V}P_v|>0$.

By Proposition 7.2, to decide whether $|\bigcap_{v\in V}P_v|>0$ in time $\mathcal{O}(2^{|V|}\cdot\text{poly}(|V|))$ and polynomial space, it is sufficient to show that for each subset $F\subseteq V$, $|\bigcap_{v\in F}\bar{P}_v|$ can be computed in polynomial time. To this end, we fix a subset $F\subseteq V$. Note that $\bigcap_{v\in F}\bar{P}_v$ is the set of all tuples $(C_1,\ldots,C_k,w_1^1,\ldots,w_k^1,w_1^2,\ldots,w_k^2,L)\in U$ such that $(\bigcup_{i=1}^k V(C_i))\cup L\subseteq V\setminus F$. Now, given an integer $\ell\in\{2k,\ldots,|V|\}$,[7] let Q_ℓ denote the set of all tuples $(C_1,\ldots,C_k,w_1^1,\ldots,w_k^1,w_1^2,\ldots,w_k^2)$ such that each C_i is a closed walk in $G-F$ of length at least three, w_i^1 and w_i^2 are consecutive occurrences of vertices in C_i, and $(\sum_{i=1}^k|V(C_i)|)=\ell$. Then, $|\bigcap_{v\in F}\bar{P}_v|=\sum_{\ell=2k}^{|V|}(|Q_\ell|\cdot\binom{|V\setminus F|}{|V|-\ell})$, where if $|V|-\ell<|V\setminus F|$, we let $\binom{|V\setminus F|}{|V|-\ell}=0$. Thus, it remains to show that each $|Q_\ell|$ can be computed in polynomial time. To this end, fix an integer $\ell\in\{2k,\ldots,|V|\}$.

Next, we will compute $|Q_\ell|$ by simply employing the method of dynamic programming. We use a matrix M that has an entry $[i,j,v,u]$ for all $i\in\{1,\ldots,k\}$, $j\in\{1,\ldots,\ell\}$, and $v,u\in V\setminus F$. Given $i\in\{1,\ldots,k\}$, $j\in\{1,\ldots,\ell\}$, and $v,u\in V\setminus F$, let $S(i,j,v,u)$ be the set of all tuples $(C_1,\ldots,C_i,w_1^1,\ldots,w_i^1,w_1^2,\ldots,w_i^2)$ such that for all $t\in\{1,\ldots,i-1\}$, C_t is a closed walk of length at least three and w_t^1 and w_t^2 are consecutive occurrences of vertices in this walk, C_i is a walk from v to u, $w_i^1=v$, and $\sum_{t=1}^i|V(C_t)|=j$. The entry $\mathrm{M}[i,j,v,u]$ will be used to store $|S(i,j,v,u)|$. Observe that

$$|Q_\ell|=\sum_{v\in V\setminus F}\ \sum_{u\in N(v)\setminus F}\ \sum_{w\in N(u)\setminus F}|S(k,\ell-1,v,w)|.$$

Thus, it remains to show that the entries of M can be calculated in polynomial time.

[7] We do not consider the case where $\ell<2k$ since k closed walks must overall contain at least $2k$ vertices.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

In the basis, we have the following calculations, relating to the case where $j = 1$:

- If $j = 1$ and ($i \geq 2$ or $v \neq u$): $\mathrm{M}[i, j, v, u] = 0$.
- Else if $j = 1$: $\mathrm{M}[i, j, v, u] = 1$.

Now, consider only entries where $j \geq 2$, which have not already been calculated in the basis. Then, we have the following calculations:

- If $i \geq 2, j \geq 3$ and $v = u$:

$$\mathrm{M}[i, j, v, u] = \sum_{w \in N(u) \setminus F} \mathrm{M}[i, j-1, v, w] + \sum_{p \in V \setminus F} \sum_{q \in N(p) \setminus F} \sum_{w \in N(q) \setminus F} \mathrm{M}[i-1, j-2, p, w].$$

- Else: $\mathrm{M}[i, j, v, u] = \sum_{w \in N(u) \setminus F} \mathrm{M}[i, j-1, v, w]$.

It is straightforward to verify that the calculations are correct. The order of the computation is an ascending order with respect to j, which ensures that when an entry is calculated, the entries on which it relies have already been calculated. To output a solution, we apply a simple self-reduction from the decision to the search variant of the problem. In particular, we repeatedly remove edges until no more edges can be removed from the graph while preserving a yes-instance. ◻

We would like to mention that if one does not care about polynomial space, then Lemma 7.1 can be obtained by straightforward dynamic programming on subsets.

8. Conclusion.

In this paper we have beaten the best known $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time algorithm for Cycle Packing that is a consequence of the Erdős–Pósa theorem. For this purpose, we developed a deterministic algorithm that solves Cycle Packing in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$. Two additional advantageous properties of our algorithm are that its space complexity is polynomial in the input size and that in case a solution exists, it outputs a solution (in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$). Our technique relies on combinatorial arguments that may be of independent interest. These arguments allow us to translate any input instance of Cycle Packing into $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ instances of Cycle Packing whose sizes are small and can therefore be solved efficiently.

It remains an intriguing open question to discover the "true" running time, under reasonable complexity-theoretic assumptions, in which one can solve Cycle Packing on general graphs. In particular, we would like to pose the following question: Does there exist a $2^{\mathcal{O}(k \log k)} \cdot |V|^{\mathcal{O}(1)}$-time algorithm for Cycle Packing? This is true for graphs of bounded maximum degree as one can easily bound the number of vertices by $\mathcal{O}(k \log k)$ and then apply Lemma 7.1. Moreover, Bodlaender, Thomassé, and Yeo [6] proved that this is also true in case one seeks k edge-disjoint cycles rather than k vertex-disjoint cycles. On the negative side, recall that (for general graphs) the bound $f(k) = \mathcal{O}(k \log k)$ in the Erdős–Pósa theorem is essentially tight, and that it is unlikely that Cycle Packing is solvable in time $2^{o(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ [12], unless ETH fails. However, we do not rule out the existence of an algorithm solving Cycle Packing in time $2^{\mathcal{O}(fvs)} \cdot |V|^{\mathcal{O}(1)}$. Thus, the two most natural attempts to obtain a $2^{\mathcal{O}(k \log k)} \cdot |V|^{\mathcal{O}(1)}$-time algorithm—either replacing the bound $\mathcal{O}(k \log k)$ in the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Erdős–Pósa theorem by $\mathcal{O}(k)$ or speeding-up the computation based on DP to run in time $2^{\mathcal{O}(tw)} \cdot |V|^{\mathcal{O}(1)}$—lead to a dead end.

Acknowledgments. We would like to thank the reviewers for several suggestions and insightful remarks that have improved the presentation of the paper.

REFERENCES

[1] N. Alon, S. Hoory, and N. Linial, *The Moore bound for irregular graphs*, Graphs Combin., 18 (2002), pp. 53–57, https://doi.org/10.1007/s003730200002.

[2] R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth, *Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference*, SIAM J. Comput., 27 (1998), pp. 942–959.

[3] H. L. Bodlaender, *On disjoint cycles*, Internat. J. Found. Comput. Sci., 5 (1994), pp. 59–68.

[4] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch, *Kernel bounds for path and cycle problems*, Theoret. Comput. Sci., 511 (2013), pp. 117–136, https://doi.org/10.1016/j.tcs.2012.09.006.

[5] H. L. Bodlaender, E. Penninkx, and R. B. Tan, *A linear kernel for the k-disjoint cycle problem on planar graphs*, in Proceedings of the 19th International Symposium on Algorithms and Computation, Gold Coast, Australia, Lecture Notes in Comput. Sci. 5369, S. Hong, H. Nagamochi, and T. Fukunaga, eds., Springer, New York, 2008, pp. 306–317, https://doi.org/10.1007/978-3-540-92182-0_29.

[6] H. L. Bodlaender, S. Thomassé, and A. Yeo, *Kernel bounds for disjoint cycles and disjoint paths*, Theoret. Comput. Sci., 412 (2011), pp. 4570–4578, https://doi.org/10.1016/j.tcs.2011.04.039.

[7] C. Chekuri and J. Chuzhoy, *Large-treewidth graph decompositions and applications*, in Proceedings of the Symposium on Theory of Computing Conference, Palo Alto, CA, D. Boneh, T. Roughgarden, and J. Feigenbaum, eds., ACM, New York, 2013, pp. 291–300, https://doi.org/10.1145/2488608.2488645.

[8] C. Chekuri and J. Chuzhoy, *Polynomial bounds for the grid-minor theorem*, J. ACM, 63 (2016).

[9] J. Chuzhoy, *Excluded grid theorem: Improved and simplified*, in Proceedings of the 47th Annual ACM on Symposium on Theory of Computing, Portland, OR, R. A. Servedio and R. Rubinfeld, eds., ACM, New York, 2015, pp. 645–654, https://doi.org/10.1145/2746539.2746551.

[10] J. Chuzhoy, *Excluded grid theorem: Improved and simplified*, in Proceedings of the 15th Scandinavian Symposium and Workshops on Algorithm Theory, Reykjavik, Iceland, R. Pagh, ed., LIPIcs 53, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016, https://doi.org/10.4230/LIPIcs.SWAT.2016.31.

[11] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, New York, 2015.

[12] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk, *Solving connectivity problems parameterized by treewidth in single exponential time*, in Proceedings of the 52nd Annual Symposium on Foundations of Computer Science, Palm Springs, CA, R. Ostrovsky, ed., IEEE Computer Society, Washington, DC, 2011, pp. 150–159, https://doi.org/10.1109/FOCS.2011.23.

[13] R. Diestel, *Graph Theory*, 4th ed., Grad. Texts in Math. 173, Springer, New York, 2012.

[14] F. Dorn, F. V. Fomin, and D. M. Thilikos, *Catalan structures and dynamic programming in h-minor-free graphs*, J. Comput. System Sci., 78 (2012), pp. 1606–1622, https://doi.org/10.1016/j.jcss.2012.02.004.

[15] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Springer, New York, 2013.

[16] P. Erdős and L. Pósa, *On independent circuits contained in a graph*, Canad. J. Math., 17 (1965), pp. 347–352.

[17] F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh, *Bidimensionality and EPTAS*, in Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, D. Randall, ed., SIAM, Philadelphia, 2011, pp. 748–759, https://doi.org/10.1137/1.9781611973082.59.

[18] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos, *Bidimensionality and kernels*, in Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algo-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

rithms, Austin, TX, M. Charikar, ed., SIAM, Philadelphia, 2010, pp. 503–510, https://doi.org/10.1137/1.9781611973075.43.

[19] F. V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi, *Kernelization: Theory of Parameterized Preprocessing*, Cambridge University Press, Cambridge, 2018.

[20] Z. Friggstad and M. R. Salavatipour, *Approximability of packing disjoint cycles*, Algorithmica, 60 (2011), pp. 395–400, https://doi.org/10.1007/s00453-009-9349-5.

[21] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Vol. 29, W. H. Freeman, New York, 2002.

[22] M. Grohe and M. Grüber, *Parameterized approximability of the disjoint cycle problem*, in Proceedings of the 34th International Colloquium Automata, Languages and Programming, Wroclaw, Poland, L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, eds., Lecture Notes in Comput. Sci. 4596, Springer, New York, 2007, pp. 363–374, https://doi.org/10.1007/978-3-540-73420-8_33.

[23] R. Impagliazzo and R. Paturi, *On the complexity of k-SAT*, J. Comput. Syst. Sci., 62 (2001), p. 367–375.

[24] A. Itai and M. Rodeh, *Finding a minimum circuit in a graph*, SIAM J. Comput., 7 (1978), pp. 413–423, https://doi.org/10.1137/0207033.

[25] F. Joos, *Parity linkage and the Erdős-Pósa property of odd cycles through prescribed vertices in highly connected graphs*, in Proceedings of Graph-Theoretic Concepts in Computer Science—41st International Workshop, Garching, Germany, E. W. Mayr, ed., Lecture Notes in Comput. Sci. 9224, Springer, New York, 2015, pp. 339–350, https://doi.org/10.1007/978-3-662-53174-7_24.

[26] N. Kakimura and K. Kawarabayashi, *Half-integral packing of odd cycles through prescribed vertices*, Combinatorica, 33 (2013), pp. 549–572, https://doi.org/10.1007/s00493-013-2865-6.

[27] N. Kakimura, K. Kawarabayashi, and D. Marx, *Packing cycles through prescribed vertices*, J. Combin. Theory Ser. B, 101 (2011), pp. 378–381, https://doi.org/10.1016/j.jctb.2011.03.004.

[28] K. Kawarabayashi and Y. Kobayashi, *Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem*, J. Combin. Theory Ser. B, 102 (2012), pp. 1020–1034, https://doi.org/10.1016/j.jctb.2011.12.001.

[29] K. Kawarabayashi, D. Král', M. Krcál, and S. Kreutzer, *Packing directed cycles through a specified vertex set*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, S. Khanna, ed., SIAM, Philadelphia, 2013, pp. 365–377, https://doi.org/10.1137/1.9781611973105.27.

[30] K. Kawarabayashi and A. Nakamoto, *The Erdős-Pósa property for vertex- and edge-disjoint odd cycles in graphs on orientable surfaces*, Discrete Math., 307 (2007), pp. 764–768, https://doi.org/10.1016/j.disc.2006.07.008.

[31] K. Kawarabayashi and B. A. Reed, *Highly parity linked graphs*, Combinatorica, 29 (2009), pp. 215–225.

[32] K. Kawarabayashi and P. Wollan, *Non-zero disjoint cycles in highly connected group labelled graphs*, J. Combin. Theory Ser. B, 96 (2006), pp. 296–301, https://doi.org/10.1016/j.jctb.2005.08.001.

[33] M. Krivelevich, Z. Nutov, M. R. Salavatipour, J. Yuster, and R. Yuster, *Approximation algorithms and hardness results for cycle packing problems*, ACM Trans. Algorithms, 3 (2007), https://doi.org/10.1145/1290672.1290685.

[34] D. Lokshtanov, F. Panolan, M. S. Ramanujan, and S. Saurabh, *Lossy kernelization*, in Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, Montreal, QC, Canada, 2017, pp. 224–237, https://doi.org/10.1145/3055399.3055456.

[35] J. Nederlof, *Fast polynomial-space algorithms using inclusion-exclusion*, Algorithmica, 65 (2013), pp. 868–884, https://doi.org/10.1007/s00453-012-9630-x.

[36] M. Pontecorvi and P. Wollan, *Disjoint cycles intersecting a set of vertices*, J. Combin. Theory Ser. B, 102 (2012), pp. 1134–1141, https://doi.org/10.1016/j.jctb.2012.05.004.

[37] V. Raman, S. Saurabh, and C. R. Subramanian, *Faster fixed parameter tractable algorithms for finding feedback vertex sets*, ACM Trans. Algorithms, 2 (2006), pp. 403–415, https://doi.org/10.1145/1159892.1159898.

[38] D. Rautenbach and B. A. Reed, *The Erdős-Pósa property for odd cycles in highly connected graphs*, Combinatorica, 21 (2001), pp. 267–278, https://doi.org/10.1007/s004930100024.

[39] B. A. Reed, *Mangoes and blueberries*, Combinatorica, 19 (1999), pp. 267–296, https://doi.org/10.1007/s004930050056.

[40] N. Robertson and P. D. Seymour, *Graph minors. V. Excluding a planar graph*, J. Combin. Theory Ser. B, 41 (1986), pp. 92–114, https://doi.org/10.1016/0095-8956(86)90030-4.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[41] N. Robertson and P. D. Seymour, *Graph minors. XIII. The disjoint paths problem*, J. Combin. Theory, Ser. B, 63 (1995), pp. 65–110, https://doi.org/10.1006/jctb.1995.1006.

[42] A. Slivkins, *Parameterized tractability of edge-disjoint paths on directed acyclic graphs*, SIAM J. Discrete Math., 24 (2010), pp. 146–157, https://doi.org/10.1137/070697781.

[43] C. Thomassen, *The Erdős-Pósa property for odd cycles in graphs of large connectivity*, Combinatorica, 21 (2001), pp. 321–333, https://doi.org/10.1007/s004930100028.

[44] J. M. M. van Rooij, M. E. van Kooten Niekerk, and H. L. Bodlaender, *Partition into triangles on bounded degree graphs*, Theory Comput. Syst., 52 (2013), pp. 687–718.

Downloaded 03/05/20 to 129.177.94.75. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.